



(12)

EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention
 of the grant of the patent:
 08.10.2003 Bulletin 2003/41

(51) Int Cl.7: G06K 9/66

(21) Application number: 01125305.1

(22) Date of filing: 18.06.1993

(54) A training method and apparatus for creating a new neuron

Trainingsverfahren und Gerät zur Erzeugung eines neuen Neurons

Méthode et appareil d'entraînement pour créer un nouveau neurone

(84) Designated Contracting States:
 AT BE CH DE DK ES FR GB GR IE IT LI LU MC NL
 PT SE

(30) Priority: 19.06.1992 US 901429

(43) Date of publication of application:
 17.04.2002 Bulletin 2002/16

(62) Document number(s) of the earlier application(s) in
 accordance with Art. 76 EPC:
 93109762.0 / O 574 936

(73) Proprietor: UNITED PARCEL SERVICE OF
 AMERICA, INC.
 Atlanta, GA 30346 (US)

(72) Inventors:
 • Moed, Michael C.
 Norwalk, Connecticut 06851 (US)
 • Lee, Chih-Ping
 c/o United Parcel Serv.America Inc
 Atlanta, Georgia 30328 (US)

(74) Representative: Beetz & Partner Patentanwälte
 Steinsdorferstrasse 10
 80538 München (DE)

(56) References cited:

- YOSHIKAZU MIYANAGA ET AL: "PARALLEL AND ADAPTIVE CLUSTERING METHOD SUITABLE FOR A VLSI SYSTEM" SIGNAL IMAGE AND VIDEO PROCESSING, SINGAPORE, JUNE 11 -14, 1991, PROCEEDINGS OF THE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, NEW YORK, IEEE, US, vol. 1 SYMP, 24, 11 June 1991 (1991-06-11), pages 356-359, XP000384785 ISBN: 0-7803-0050-5
- KAVURI S N ET AL: "Solving the hidden node problem in networks with ellipsoidal units and related issues" PROCEEDINGS OF THE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS. (IJCNN), BALTIMORE, JUNE 7 - 11, 1992, NEW YORK, IEEE, US, vol. 3, 7 June 1992 (1992-06-07), pages 775-780, XP010060146 ISBN: 0-7803-0559-0
- KELLY P ET AL: "An adaptive algorithm for modifying hyperellipsoidal decision surfaces" PROCEEDINGS OF THE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS. (IJCNN), BALTIMORE, JUNE 7 - 11, 1992, NEW YORK, IEEE, US, vol. 3, 7 June 1992 (1992-06-07), pages 196-201, XP010060109 ISBN: 0-7803-0559-0

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

Description

[0001] The present invention relates to classification methods and systems, and, in particular, to methods and systems for classifying optically acquired character images and to methods and systems for training such.

5

2. Statement of Related Art

[0002] In the document YOSHIKAZU MIYANAGA ET AL: 'PARALLEL AND ADAPTIVE CLUSTERING METHOD SUITABLE FOR A VLSI SYSTEM' SIGNAL IMAGE AND VIDEO PROCESSING, SINGAPORE, JUNE 11 -14, 1991, PROCEEDINGS OF THE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, NEW YORK, IEEE, US, vol. 1, SYMP. 24, 11 June 1991 (1991-06-11), pages 356-359, XP000384785 ISBN: 0-7803-0050-5 a two-functional network is proposed in which adaptive methods are implemented for sophisticated recognition and clustering. In the first subnetwork, self-organized clustering is realized. The clustering is based on Mahalanobis distance. The result of the first subnetwork becomes a vector of similarity values between a given input pattern and all patterns of cluster nodes. The second subnetwork determines the optimum label from the similarity network. The second network consists of nodes associated with specific labels. All connections between the label nodes of the second functional network and the cluster nodes of the first functional network are determined by supervised learning. Every calculation is executed in parallel and pipelined forms. Figure 3 shows the block diagram of the control flow. An input pattern is fed into all nodes of the first sub-network at each instance. At the fist step the similarity values of all nodes are calculated. If a new node should be created, an input pattern is assigned to the new node. If it is not necessary to create a new node, the optimum node is selected from the similarity values.

15

[0003] In the field of package shipping, packages are routed from origins to destinations throughout the world according to destination addresses typed on shipping labels applied to these packages. In order to route packages, it is desirable to use automated optical character classification systems that can read those addresses. Such a classification system must be able to classify characters as quickly as possible. Conventional optical character classification systems using spherical neurons, such as those disclosed in U.S. Patent No. 4,328,259 (Cooper et al.), may be unable to execute the processing requirements presented by certain applications without a substantial investment in hardware.

25

[0004] In the document KELLY P M ET AL: 'An adaptive algorithm for modifying hyperellipsoidal decision surfaces' PROCEEDINGS OF THE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS. (IJCNN), BALTIMORE, JUNE 7 - 11, 1992, NEW YORK, IEEE, US, vol. 3, 7 June 1992 (1992-06-07), pages 198-201, XP010060109 ISBN: 0-7803-0559-0 a learning algorithm for a distance classifier has been developed which manipulates hyperellipsoidal cluster boundaries. Regions of the input feature space are first enclosed by ellipsoidal boundaries, and then these boundaries are iteratively modified to reduce classification error in areas of known overlap between different classes. During adaption each hyperellipsoid in the classifier will maintain its original orientation, although its position and shape will be modified. The algorithm that is used is referred as the LVQ-MM algorithm (LVQ with the Mahalanobis distance Metric).

35

SUMMARY OF THE INVENTION

40

[0005] The present invention covers a training method and apparatus for creating a new neuron in a feature space having at least one existing neuron. The invention generates a feature vector representative of a training input, where the training input corresponds to one of a plurality of possible outputs. If no existing neuron corresponding to the training input encompasses the feature vector, then the invention creates a new neuron, where the new neuron comprises a boundary defined by two or more neuron axes of different length.

45

BRIEF DESCRIPTION OF THE DRAWINGS**[0006]**

50

Figs. 1(a), 1(b), 1(c), and 1(d) are bitmap representations of a nominal letter "O", a degraded letter "O", a nominal number "7", and a degraded letter "7", respectively;

Fig. 2 is a graphical depiction of a 2-dimensional feature space populated with 8 elliptical neurons that may be employed by the classification system of the present invention to classify images of the letters A, B, and C;

55

Fig. 3 is a process flow diagram for classifying inputs according to a preferred embodiment of the present invention;

Fig. 4 is a schematic diagram of part of the classification system of Fig. 3;

Fig. 5 is a process flow diagram for generating neurons used by the classification system of Fig. 3; and

Fig. 6 is a schematic diagram of a classification system that uses cluster classifiers for classifying inputs according to a preferred embodiment of the present invention.

5 DETAILED DESCRIPTION OF THE INVENTION

[0007] The present invention includes a system for optical character recognition, but, more generally, the invention covers a classification system for classifying an input as one of a defined set of possible outputs. For example, where the input is an optically acquired image representing one of the 26 capital letters of the English alphabet, the classification system of the present invention may be used to select as an output that capital letter that is associated with the input image. The classification system of the present invention is discussed below in connection with Figs. 1(a), 2, 3, and 4.

[0008] The present invention also includes a system for "training" the classification system of the present invention. This training system is preferably operated off line prior to deployment of the classification system. In the character recognition example, the training system accepts input images representative of known characters to "learn" about the set of possible outputs into which unknown images will eventually be classified. The training system of the present invention is discussed below in connection with Fig. 5.

[0009] The present invention also includes a system for training the classification system of the present invention based on ordering the training inputs according to the relative quality of the training inputs. This system for training is discussed below in connection with Figs. 1(a), 1(b), 1(c), and 1(d).

[0010] The present invention also includes a system for adjusting the locations and shapes of neurons generated during the training systems of the present invention.

[0011] The present invention also includes a classification system employing a hierarchical network of top-level and lower level cluster classifiers. The top-level classifier classifies inputs into one of a plurality of output clusters, where each output cluster is associated with a subset_of the set of possible outputs. A cluster classifier, associated with the output cluster identified by the top-level classifier, then classifies the input as corresponding to one of the possible outputs. This classification system is discussed below in connection with Fig. 1(a), 1(b), 1(c), and 1(d).

[0012] The present invention also includes a neural system of classifying inputs that combines two subsystems. One subsystem counts the number of neurons that encompass a feature vector representing a particular input for each of the possible outputs. If one of the possible outputs has more neurons encompassing the feature vector than any other possible output, then the system selects that possible output as corresponding to that input. Otherwise, the second subsystem finds the neuron that has the smallest value for a particular distance measure for that feature vector. If that value is less than a specified threshold then the system selects the output associated with that neuron as corresponding to the input. This neural system is discussed below in connection with Figs. 1(a), 2, 3, and 4.

CLASSIFICATION SYSTEM

[0013] Referring now to Fig. 1(a), there is shown a bitmap representation of a nominal letter "O". When the classification system of the present invention classifies optically acquired character images, each character image to be classified may be represented by an input bitmap, an $(m \times n)$ image array of binary values as shown in Fig. 1(a). In a preferred embodiment, the classification system of the present invention generates a vector in a k -dimensional feature space from information contained in each input bitmap. Each feature vector F has feature elements f_j , where $0 \leq j \leq k-1$. The dimension of the feature space, k , may be any integer greater than one. Each feature element f_j is a real value corresponding to one of k features derived from the input bitmap.

[0014] The k features may be derived from the input bitmap using conventional feature extraction functions, such as, for example, the Grid or Hadamard feature extraction function. The feature vector F represents a point in the k -dimensional feature space. The feature elements f_j are the components of feature vector F along the feature-space axes of the k -dimensional feature space. For purposes of this specification, the term "feature vector" refers to a point in feature space.

[0015] In a preferred embodiment, a discriminant analysis transform may be applied to Grid-based or Hadamard-based feature vectors to define the feature space. In this embodiment, the separation between possible outputs may be increased and the dimensionality of the feature vector may be reduced by performing this discriminant analysis in which only the most significant Eigenvectors from the discriminant transformation are retained.

[0016] The classification system of the present invention, compares a feature vector F , representing a particular input image, to a set of neurons in feature space, where each neuron is a closed k -dimensional region or "hyper-volume" in the k -dimensional feature space. For example, when ($k = 2$), each neuron is an area in a 2-dimensional feature space, and when ($k = 3$), each neuron is a volume in a 3-dimensional feature space. Fig. 2 shows a graphical depiction of an

exemplary 2-dimensional feature space populated with eight 2-dimensional neurons.

[0017] In a preferred classification system according to the present invention, the boundary of at least one of the neurons populating a k -dimensional feature space is defined by at least two axes that have different lengths. Some of these neurons may be generally represented mathematically as:

$$\sum_{j=0}^{k-1} \frac{|c_j - g_j|^m}{(b_j)^m} \leq A, \quad (1)$$

where c_j define the center point of the neuron, b_j are the lengths of the neuron axes, and m and A are positive real constants. In a preferred embodiment, at least two of the neuron axis are of different length. The values g_j that satisfy Equation (1) define the points in feature space that lie within or on the boundary of the neuron. Those skilled in the art will understand that other neurons within the scope of this invention may be represented by other mathematical expressions. For example, a neuron may be defined by the expression:

$$\max_{j=0}^{k-1} \frac{|c_j - g_j|}{|b_j|} \leq 1, \quad (2)$$

where the function "MAX" computes the maximum value of the ratio as j runs from 0 to $k-1$. Neurons defined by Equation (2) are hyper-rectangles.

[0018] In a preferred embodiment of the present invention, the neurons are hyper-ellipses in the k -dimensional feature space. A hyper-ellipse is any hyper-volume defined by Equation (1), where ($m=2$) and ($A=1$). More particularly, a hyper-ellipse is defined by the function:

$$\sum_{j=0}^{k-1} \frac{(c_j - g_j)^2}{(b_j)^2} \leq 1, \quad (3)$$

where c_j define the hyper-ellipse center point, b_j are the hyper-ellipse axis lengths, and the values g_j that satisfy Equation (3) define the points that lie within or on the hyper-ellipse boundary. When all of the axes are the same length, the hyper-ellipse is a hyper-sphere. In a preferred embodiment of the present invention, in at least one of the neurons, at least two of the axes are of different length. By way of example, there is shown in Fig. 2 elliptical neuron 1, having center point (c_0^1, c_1^1) , and axis b_0^1, b_1^1 of different length. In a preferred embodiment, the axes of the neurons are aligned with the coordinate axes of the feature space. Those skilled in the art will understand that other neurons having axes that do not all align with the feature-space axes are within the scope of the invention.

[0019] According to the present invention, each neuron is associated with a particular possible output. For example, each neuron may correspond to one of the 26 capital letters. Each neuron is associated with only one of the possible outputs (e.g., letters), but each possible output may have one or more associated neurons. Furthermore, neurons may overlap one another in feature space. For example, as shown in Fig. 2, neurons 0, 1, and 7 correspond to the character "A", neurons 2, 3, 5, and 6 correspond to the character "B", and neuron 4 corresponds to the character "C". Neurons 1 and 7 overlap, as do neurons 2, 3, and 6 and neurons 3, 5, and 6. In an alternative embodiment (not shown), neurons corresponding to different possible outputs may overlap. The classification system of the present invention may employ the neurons of Fig. 2 to classify input images representative of the letters A, B, and C.

[0020] Referring now to Fig. 3, there is shown a process flow diagram of classification system 300 for classifying an input (e.g., a bitmap of an optically acquired character image) as one of a set of possible outputs (e.g., characters) according to a preferred embodiment of the present invention. In the preferred embodiment shown in Fig. 3, the neurons in classification system 300 are processed in parallel. In an alternative embodiment (not shown), the neurons of classification system 300 may be processed in series. Means 302 is provided for receiving an input image bitmap and generating a feature vector that represents information contained in that bitmap. Means 304 and 306 are provided for comparing the feature vector generated by means 302 to a set of neurons, at least one of which has two or more axes of different length. Classification system 300 selects one of the possible outputs based upon that comparison.

[0021] In a preferred embodiment of the present invention, classification system 300 classifies optically acquired character bitmaps using a network of hyper-elliptical neurons. Means 302 of classification system 300 receives as input the bitmap of an optically acquired character image to be classified and generates a corresponding feature vector F . Means 304 then determines an "elliptical distance" r_x as a function of the center and axes of each of the E_{num} hyper-elliptical neurons x in the network and feature vector F , where:

$$r_x = \sum_{j=0}^{k-1} \frac{(F_j - c_j^x)^2}{(b_j^x)^2}, \quad (4)$$

In Equation (4), c_j^x and b_j^x define the center point and axis lengths, respectively, of neuron x , where x runs from 0 to $E_{num}-1$, and F_j are the elements of feature vector F . Those skilled in the art would recognize that distance measures different from that of Equation (4) may also be used.

[0022] Means 306 determines which, if any, of the E_{num} neurons encompass feature vector F . A neuron encompasses a feature vector -- and may be referred to as an "encompassing neuron" -- if the feature vector lies inside the boundary that defines the neuron in feature space. For hyper-ellipses, neuron x encompasses feature vector F , if $(r_x < 1)$. If $(r_x = 1)$, feature vector F lies on the boundary of neuron x , and if $(r_x > 1)$, feature vector F lies outside neuron x . Since neurons may overlap in feature space, a particular feature vector may be encompassed by more than one neuron. In Fig. 2, feature vector F_g , corresponding to a particular input image, is encompassed by neurons 2 and 6. Alternatively, a feature vector may lie inside no neurons, as in the case of feature vector F_h of Fig. 2, which corresponds to a different input image.

[0023] Means 308 finds the "closest" neuron for each possible output. As described earlier, each neuron is associated with one and only one possible output, but each possible output may have one or more neurons associated with it. Means 308 analyzes all of the neurons associated with each possible output and determines the neuron "closest" to feature vector F for that output. The "closest" neuron will be the one having the smallest "distance" measure value r_x . In the example of feature vector F_g of Fig. 2, means 308 will select neuron 1 as being the "closest" neuron to feature vector F_g for the character "A". It will also select neuron 2 as the "closest" neuron for character "B" and neuron 4 for character "C".

[0024] Means 310 in Fig. 3 counts votes for each possible output. In a first preferred embodiment, each neuron that encompasses feature vector F is treated by means 310 as a single "vote" for the output associated with that neuron. In an alternative preferred embodiment discussed in greater detail with respect to Equation (7) below, each neuron that encompasses feature vector F is treated by means 310 as representing a "weighted vote" for the output associated with that neuron, where the weight associated with any particular neuron is a function of the number of training input feature vectors encompassed by that neuron. In a preferred embodiment, means 310 implements proportional voting, where the weighted vote for a particular neuron is equal to the number of feature vectors encompassed by that neuron. For each possible output, means 310 tallies all the votes for all the neurons that encompass feature vector F . There are three potential types of voting outcomes: either (1) one output character receives more votes than any other output character, (2) two or more output characters tie for the most votes, or (3) all output characters receive no votes, indicating the situation where no neurons encompass feature vector F . In Fig. 2, feature vector F_g may result in the first type of voting outcome: character "B" may receive 2 votes corresponding to encompassing neurons 2 and 6, while characters "A" and "C" receive no votes. Feature vector F_h of Fig. 2 results in the third type of voting outcome with each character receiving no votes.

[0025] Means 312 determines if the first type of voting outcome resulted from the application of means 310 to feature vector F . If only one of the possible output characters received the most votes, then means 312 directs the processing of classification system 300 to means 314, which selects that output character as corresponding to the input character bitmap. Otherwise, processing continues to means 316. For feature vector F_g in Fig. 2, means 312 determines that character "B" has more votes than any other character and directs means 314 to select "B" as the character corresponding to feature vector F_g . For feature vector F_h in Fig. 2, means 312 determines that no single character received the most votes and directs processing to means 316.

[0026] Means 316 acts as a tie-breaker for the second and third potential voting outcome in which no outright vote-leader exists, either because of a tie or because the feature vector lies inside no neurons. To break the tie, means 316 selects that neuron x which is "closest" in elliptical distance to feature vector F and compares r_x to a specified threshold value θ^* . If $(r_x \leq \theta^*)$, then means 318 selects the output character associated with neuron x as corresponding to the input character bitmap. Otherwise, the tie is not broken and classification system 300 selects no character for the input image. A "no-character-selected" result is one of the possible outputs from classification system 300. For example, if

classification system 300 is designed to recognize capital letters and the input image corresponds to the number "7", a no-character-selected result is an appropriate output.

[0027] Threshold value θ^m may be any number greater than 1 and is preferably about 1.25. As described earlier, when feature vector F is inside neuron x , then ($r_x < 1$), and when feature vector F is outside neuron x , then ($r_x > 1$). If the voting result from means 310 is a tie for the most non-zero votes, then means 316 will select the output character associated with the encompassing neuron having a center which is "closest" in elliptical "distance" feature vector F . Alternatively, if there are no encompassing neurons, means 316 may still classify the input bitmap as corresponding to the output character associated with the "closest" neuron X , then ($r_x \leq \theta^m$). Using a threshold value θ^m of about 1.25 establishes a region surrounding each neuron used by means 316 for tie-breaking. In Fig. 2, feature vector F_h will be classified as character "C" if the "distance" measure r_h is less than the threshold value θ^m ; otherwise, no character is selected.

[0028] Referring now to Fig. 4, there is shown a schematic diagram of classification system 400 of the present invention for classifying inputs as corresponding to a set of s possible outputs. Classification system 400 may perform part of the processing performed by classification system 300 of Fig. 3. Classification system 400 accepts feature vector F , represented by feature elements $(f_0, f_1, \dots, f_{k-1})$, and generates values c^d and c^a that act as pointers and/or flags to indicate the possible output to be selected. Classification system 400 includes four subsystem levels: input level 402, processing level 404, output level 406, and postprocessing level 408.

[0029] Input level 402 includes the set of k input processing units i_j , where j runs from 0 to $k-1$. Each input processing unit i_j receives as input one and only one element f_j of the feature vector F and broadcasts this value to processing level 404. Input level 402 functions as a set of pass-through, broadcasting elements.

[0030] Processing level 404 includes the set E of E_{num} elliptical processing units e_x , where x runs from 0 to $E_{num}-1$. Each elliptical processing unit e_x is connected to and receives input from the output of every input processing unit i_j of input level 402. Elliptical processing unit e_x implements Equation (4) for neuron x of classification system 300 of Fig. 3. Like neuron x of classification system 300, each elliptical processing unit e_x is defined by two vectors of internal parameters: B^x and C^x . The elements of vector B^x are the lengths of the axes of neuron x , where:

$$B^x = (b_0^x, b_1^x, \dots, b_{k-1}^x)^T \quad (5)$$

and the elements of vector C^x are the coordinates of the center point of neuron x , where:

$$C^x = (c_0^x, c_1^x, \dots, c_{k-1}^x)^T \quad (6)$$

[0031] Each elliptical processing unit e_x of processing level 404 computes the distance measure r_x from feature vector F to the center of neuron x . Processing level 404 is associated with means 304 of classification system 300. If ($r_x < 1$), then elliptical processing unit e_x is said to be activated; otherwise, elliptical processing unit e_x is not activated. In other words, elliptical processing unit e_x is activated when neuron x encompasses feature vector F . Each elliptical processing unit e_x broadcasts the computed distance measure r_x to only two output processing units of output level 406.

[0032] Output level 406 includes two parts: output-total part 410 and output-minimize part 412. Output-total part 410 contains the set O of s output processing units o_n^t , and output-minimize part 412 contains the set O^m of s output processing units o_n^m , where n runs from 0 to $s-1$, where s is also the number of possible outputs for which classification system 400 has been trained. For example, when classifying capital letters, $s=26$. Each processing unit pair (o_n^t, o_n^m) is associated with only one possible output and vice versa.

[0033] Each elliptical processing unit e_x of processing level 404 is connected to and provides output to only one output processing unit o_n^t of output-total part 410 and to only one output processing unit o_n^m of output-minimize part 412. However, each output processing unit o_n^t and each output processing unit o_n^m may be connected to and receive input from one or more elliptical processing units e_x of processing level 404. These relationships are represented by connection matrices W and W^m , both of which are of dimension $(s \times E_{num})$. In a preferred embodiment, if there is a connection between elliptical processing unit e_x of processing level 404 and output processing unit o_n^t of output-total part 410 of output level 406, an entry w_{nx}^t in connection matrix W^t will have a value that is equal to the number of training input feature vectors encompassed by neuron x ; otherwise, it has value 0. In a further preferred embodiment, entry w_{nx}^t has a value 1 if there is a connection between elliptical processing unit e_x and output processing unit O_n^t .

[0034] Connection matrix W^m represents the connections between processing level 404 and output-minimize part 412 of output level 406 and is related to connection matrix W^t . An entry w_{nx}^m in connection matrix W^m will have a value of 1 for every entry w_{nx}^t in connection matrix W^t that is not zero. Otherwise, entry w_{nx}^m will have a value of 0.

[0035] Each output processing unit o_n^t in output-total part 410 computes an output value O_n^t , where:

$$o_n^t = \sum_{x=0}^{E_{\text{num}}-1} w_{nx}^t T(r_x), \quad (7)$$

5

where the function $T(r_x)$ returns the value 0 if ($r_x > 1$); otherwise, it returns the value 1. In other words, the function $T(r_x)$ returns the value 1 if elliptical processing unit e_x of processing level 404 is activated. Output processing unit O_n^t counts the votes for the possible output with which it is associated and outputs the total. Output-total part 410 of output level 406 is associated with means 306 and means 310 of classification system 300.

[0036] Similarly, each output processing unit O_n^m in output-minimize part 412 computes an output value O_n^m , where:

$$O_n^m = \min_{x=0}^{E_{\text{num}}-1} (w_{nx}^m r_x), \quad (8)$$

for all $w_{nx}^m \neq 0$

20

where the function "MIN" returns the minimum value of ($w_{nx}^m r_x$) over all the elliptical processing units e_x . Therefore, each output processing unit O_n^m examines each of the elliptical processing units e_x to which it is connected and outputs a real value equal to the minimum output value from these elliptical processing units. Output-minimize part 412 of output level 406 is associated with means 308 of classification system 300.

[0037] Postprocessing level 408 includes two postprocessing units p^d and p^m . Postprocessing unit p^d is connected to and receives input from every output processing unit O_n^t of output-total part 410 of output level 406. Postprocessing unit p^d finds the output processing unit O_n^t that has the maximum output value and generates the value q^d . If output processing unit O_n^t of output-total part 410 has an output value greater than those of all the other output processing units of output-total part 410, then the value q^d is set to $n - 1$ -- the Index for that output processing unit. For example, when classifying capital letters n may be O for "A" and 1 for "B", etc. Otherwise, the value q^d is set to -1 to indicate that output-total part 410 of output level 406 did not classify the input. Postprocessing unit p^d of postprocessing level 408 is associated with means 312 of classification system 300.

[0038] Similarly, postprocessing unit p^m -- the other postprocessing unit in postprocessing level 408 -- is connected to and receives input from every output processing unit O_n^m of output-minimize part 412 of output level 406. Postprocessing unit p^m finds the output processing unit O_n^m that has the minimum output value and generates the value q^m . If output processing unit O_n^m of output-minimize part 412 has an output value less than a specified threshold θ^m , then the value q^m is set to the corresponding index n . Otherwise, the value q^m is set to -1 to indicate that output-minimize part 412 of output level 406 did not classify the input, because the feature vector F is outside the threshold region surrounding neuron x for all neurons x . The threshold θ^m may be the same threshold θ^m used in classification system 300 of Fig. 3. Postprocessing unit P^m of postprocessing level 408 is associated with means 316 of classification system 300.

[0039] Classification of the input is completed by analyzing the values q^d and q^m . If ($q^d \neq -1$), then the input is classified as possible output q^d of the set of s possible outputs. If ($q^d = -1$) and ($q^m \neq -1$), then the input is classified as possible output q^m of the set of s possible outputs. Otherwise, if both values are -1, then the input is not classified as any of the s possible outputs.

TRAINING SYSTEM

[0040] A neural network must be trained before it may be used to classify inputs. The training system of the present invention performs this required training by generating at least one non-spherical neuron in the K -dimensional feature space. The training system is preferably implemented off line prior to the deployment of a classification system.

[0041] The training system of the present invention generates neurons based upon a set of training inputs, where each training input is known to correspond to one of the possible outputs in the classification set. Continuing with the example of capital letters used to describe classification system 300, each training input may be a bitmap corresponding to one of the characters from "A" to "Z". Each character must be represented by at least one training input, although typically 250 to 750 training inputs are used for each character.

[0042] Referring now to Fig. 5, there is shown a process flow diagram of training system 500 for generating neurons in K -dimensional feature space that may be used in classification system 300 of Fig. 3 or in classification system 400

of Fig. 4. For example, when training for output classification, training system 500 sequentially processes a set of training bitmap inputs corresponding to known outputs. At a particular point in the training, there will be a set of existing feature vectors that correspond to the training inputs previously processed and a set of existing neurons that have been generated from those existing feature vectors. For each training input, training system 500 generates a feature vector in a feature space that represents information contained in that training input.

[0043] Training system 500 applies two rules in processing each training input. The first training rule is that if the feature vector, corresponding to the training input currently being processed, is encompassed by any existing neurons that are associated with a different known output, then the boundaries of those existing neurons are spatially adjusted to exclude that feature vector – that is, to ensure that that feature vector is not inside the boundary of those existing neurons. Otherwise, neurons are not spatially adjusted. For example, if the current training input corresponds to the character "R" and the feature vector corresponding to that training input is encompassed by two existing "P" neurons and one existing "B" neuron, then the boundaries of these three existing neurons are spatially adjusted to ensure they do not encompass the current feature vector.

[0044] The second training rule is that if the current feature vector is not encompassed by at least one existing neuron that is associated with the same known output, then a new neuron is created. Otherwise, no new neuron is created for the current feature vector. For example, if the current training input corresponds to the character "W" and the feature vector corresponding to that training input is not encompassed by any existing neuron that is associated with the character "W", then a new "W" neuron is created to encompass that current feature vector. In a preferred embodiment, a new neuron is created by generating a temporary hyper-spherical neuron and then spatially adjusting that temporary neuron to create the new neuron. In an alternative preferred embodiment, the temporary neuron may be a non-spherical hyper-ellipse.

[0045] In a preferred embodiment of the present invention, training system 500 generates hyper-elliptical neurons from a set of training bitmap inputs corresponding to known characters. Training system 500 starts with no existing feature vectors and no existing neurons. Processing of training system 500 begins with means 502 which selects as the current training input a first training input from a set of training inputs. Means 504 generates the feature vector F that corresponds to the current training input.

[0046] When the first training input is the current training input, there are no existing neurons and therefore no existing neurons that encompass feature vector F . In that case, processing of training system 500 flows to means 514 which creates a new neuron centered on feature vector F . The new neuron is preferably defined by Equation (3), where all the new neuron axes are set to the same length, that is, $(b_j - \lambda)$ for all j . Since the new neuron axes are all the same length, the new neuron is a hyper-sphere in feature space of radius λ . In a preferred embodiment, the value of constant λ may be twice as large as the largest feature element f_j of all the feature vectors F for the entire set of training inputs. Since there are no existing feature vectors when processing the first training input, training system 500 next flows to means 528 from which point the processing of training system 500 may be described more generally.

[0047] Means 528 determines whether the current training input is the last training input in the set of training inputs. If not, then means 528 directs processing of training system 500 to means 530 which selects the next training input as the current training input. Means 504 then generates the feature vector F corresponding to the current training input.

[0048] Means 506 and 508 determine which, if any, existing neurons are to be spatially adjusted to avoid encompassing feature vector F . In a preferred embodiment, means 510 adjusts an existing neuron if that neuron is not associated with the same known character as the current training input (as determined by means 506) and if it encompasses feature vector F (as determined by means 508). Means 508 determines if an existing neuron encompasses feature vector F by calculating and testing the "distance" measure r_x of Equation (4) and testing whether ($r_x < 1$) as described earlier.

[0049] In a preferred embodiment, means 510 spatially adjusts an existing neuron by optimally shrinking it along only one axis. In another preferred embodiment, means 510 shrinks an existing neuron proportionally along 'one or more' axes. These shrinking methods are explained in greater detail later in this specification. After processing by means 510, the current feature vector is not encompassed by any existing neurons that are associated with a character which is different from the character associated with the training input. Hence, the current feature vector lies either outside or on the boundaries of such existing neurons.

[0050] Training system 500 also determines if a new neuron is to be created and, if so, creates that new neuron. A new neuron is created (by means 514) if the feature vector F is not encompassed by any existing neuron associated with the same character as the training input (as determined by means 512). As described above, means 514 creates a new neuron that is, preferably, a hyper-sphere of radius λ .

[0051] Training system 500 then tests and, if necessary, spatially adjusts each new neuron created by means 514 to ensure that it does not encompass any existing feature vectors that are associated with a character which is different from the character associated with the training input. Means 516, 524, and 526 control the sequence of testing a new neuron against each of the existing feature vectors by selecting one of the existing feature vectors at a time. If a new neuron is associated with a character different from that of the currently selected existing feature vector (as determined

by means 518) and if the new neuron encompasses that selected existing feature vector (as determined by means 520 using Equation (4)), then means 524 spatially adjusts the new neuron by one of the same shrinking algorithms employed by means 510. Training system 500 continues to test and adjust a new neuron until all existing feature vectors have been processed. Since the hyper-spherical neuron created by means 514 is adjusted by means 522, that hyper-spherical neuron is a temporary neuron with temporary neuron axes of equal length. Processing of training system 500 then continues to means 528 to control the selection of the next training input.

[0052] In a preferred embodiment, the steps of (1) shrinking existing neurons for a given input, and (2) creating and shrinking a new neuron created for that same input may be performed in parallel. Those skilled in the art will understand that these two steps may also be performed sequentially in either order.

[0053] In a preferred embodiment, after all of the training inputs in the set of training inputs have been processed sequentially, means 528 directs processing of training system 500 to means 532. After processing a set of training inputs with their corresponding feature vectors, feature space is populated with both feature vectors and neurons. After processing the set of training inputs one time, some feature vectors may not be encompassed by any neurons. This occurs when feature vectors, that were, at some point in the training process, encompassed by neuron(s) of the same character, become excluded from those neurons when those neurons were shrunk to avoid subsequent feature vectors associated with a different character. In such a situation, means 532 directs processing to return to means 502 to repeat processing of the entire set of training inputs. When repeating this processing, the previously created neurons are retained. By iteratively repeating this training process, new neurons are created with each iteration until eventually each and every feature vector is encompassed by one or more neurons that are associated with the proper output and no feature vectors are encompassed by neurons associated with different possible outputs. Moreover, this iterative training is guaranteed to converge in a finite period of time with the maximum number of iterations being equal to the total number of training inputs.

[0054] After training system 500 completes its processing, the feature space is populated with neurons that may then be used by characterization system 300 or characterization system 400 to classify an unknown input into one of a plurality of possible outputs.

OPTIMAL ONE-AXIS SHRINKING

[0055] As mentioned earlier, in a preferred embodiment, training system 500 spatially adjusts the boundary of a hyper-elliptical neuron to exclude a particular feature vector by optimally shrinking along one axis. Means 510 and 522 of training system 500 may perform this one-axis shrinking by (1) identifying the axis to shrink, and (2) calculating the new length for that axis.

[0056] Training system 500 identifies the axis n to shrink by the formula:

35

$$n = \underset{0 \leq i \leq k-1}{\operatorname{argmax}} \left[\frac{\left(\prod_{j=0}^{k-1} b_j \right) \cdot |f_i - c_i|}{\sqrt{1 - \sum_{j=0, j \neq i}^{k-1} \frac{(f_j - c_j)^2}{b_j^2}}} \right], \quad (9)$$

45

where the function "argmax" returns the value of i that maximizes the expression in the square brackets for any i from 0 to $k-1$; c_j and b_j define the center point and axis lengths, respectively, of the neuron to be adjusted; and f_j define the feature vector to be excluded by that neuron.

[0057] Training system 500 then calculates the new length b_n for axis n by the equation:

55

$$b'_n = \frac{|f_n - c_n|}{\sqrt{1 - \sum_{\substack{j=0 \\ j \neq n}}^{k-1} \frac{(f_j - c_j)^2}{b_j^2}}} \quad (10)$$

10 In one-axis shrinking, all other axes retain their original lengths b_j .

[0058] One-axis shrinking of an original hyper-elliptical neuron according to Equations (9) and (10) results in an adjusted neuron with the greatest hyper-volume V that satisfies the following four criteria:

- 15 (1) The adjusted neuron is a hyper-ellipse;
 (2) The center point of the original neuron is the same as the center point of the adjusted neuron;
 (3) The feature vector to be excluded lies on the boundary of the adjusted neuron; and
 (4) All points within or on the boundary of the adjusted neuron lie within or on the boundary of the original neuron.

20 The hyper-volume V is defined by:

$$V = C_k \prod_{j=0}^{k-1} b_j \quad (11)$$

25 where C_k is a constant that depends on the value of k , where k is the dimension of the feature space, and b_j are the lengths of the axes defining the adjusted neuron. One-axis shrinking, therefore, provides a first method for optimally adjusting neurons according to the present invention.

PROPORTIONAL SHRINKING ALGORITHM

30 [0059] In alternative preferred embodiment, training system 500 spatially adjusts the boundary of a hyper-elliptical neuron to exclude a particular feature vector by shrinking proportionally along one or more axes. Means 510 and 522 of training system 500 may perform proportional shrinking by calculating the vector ΔB of axis length changes Δb_j where:

$$40 \Delta B = -\alpha \left[\frac{|F-C|^T}{\|F-C\|} \right] \cdot (B^T B - [E-C]^T [E-C]) - \Gamma, \quad (12)$$

45 where:

$$\Delta B = (\Delta b_0, \Delta b_1, \dots, \Delta b_{k-1}) \quad (13)$$

$$50 \frac{|F-C|^T}{\|F-C\|} = \text{Vector of Cosines}, \quad (14)$$

$$|F-C|^T = (|f_0 - c_0|, |f_1 - c_1|, \dots, |f_{k-1} - c_{k-1}|), \quad (15)$$

$$55 F = (f_0, f_1, \dots, f_{k-1})^T, \quad (16)$$

$$C = (C_0, C_1, \dots, C_{k-1})^T, \quad (17)$$

5

$$B = \begin{bmatrix} b_0 & 0 & 0 & \dots & 0 \\ 0 & b_1 & 0 & \dots & 0 \\ 0 & 0 & b_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & b_{k-1} \end{bmatrix}, \quad (18)$$

10

15

$$F = \begin{bmatrix} f_0 & 0 & 0 & \dots & 0 \\ 0 & f_1 & 0 & \dots & 0 \\ 0 & 0 & f_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & f_{k-1} \end{bmatrix}, \quad (19)$$

20

25

30

$$C = \begin{bmatrix} c_0 & 0 & 0 & \dots & 0 \\ 0 & c_1 & 0 & \dots & 0 \\ 0 & 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & c_{k-1} \end{bmatrix}, \quad (20)$$

35

40

and

$$\Gamma = (\gamma_0, \gamma_1, \dots, \gamma_{k-1}), \quad (21)$$

45

where $|f_0 - c_0|$ is the absolute value of $(f_0 - c_0)$; $\|F - C\|$ is the magnitude of the vector difference between F and C ; c_j and b_j define the center point and axis lengths, respectively, of the neuron to be adjusted; f_j are the elements of the feature vector to be excluded from that neuron; and α and γ_j may be constants. The new axis lengths b_j' for the adjusted neuron are calculated by:

50

$$b_j' = b_j + \Delta b_j \quad (22)$$

for j from 0 to $k-1$.

[0060] In proportional shrinking, training system 500 determines the projections of a vector onto the axes of the neuron to be adjusted, where the vector points from the center of that neuron to the feature vector to be excluded. These projections are represented by the vector of cosines of Equation (14). Training system 500 then determines

how much to shrink each neuron axis based on the relationship between the length of the axis and the length of the projection onto that axis.

[0061] In a preferred embodiment, the constant α in Equation (12) is selected to be less than 1. In this case, training system 500 may perform iterative shrinking, where the neuron is slowly adjusted over multiple axis-shrinking steps until it is determined that the feature vector to be excluded is outside the adjusted neuron. In a preferred embodiment, parameter ' γ ' may be set to a positive value that is roughly 0.001 times the size of axis j to ensure that proportional shrinking eventually places the feature vector outside the neuron. In an alternative preferred embodiment, the parameters γ may be error functions based on the distance from the feature vector to the boundary of the slowly adjusted neuron. In such case, training system 500 may operate as a proportional integral controller for adjusting neurons.

10

ORDERING OF TRAINING INPUTS

[0062] In a preferred embodiment of the present invention, the set of training inputs, used sequentially by the training system to generate neurons, may be organized according to input quality. The training inputs may be ordered to train with higher quality inputs before proceeding to those of lower quality. This quality ordering of training inputs ensures that neurons are centered about feature vectors that correspond to inputs of higher quality. Such ordered training may improve the performance efficiency of a classification system by reducing the numbers of neurons needed to define the classification system. Such ordering may also reduce the numbers of misclassifications and non-classifications made by the classification system. A misclassification is when a classification system selects one possible output when, in truth, the input corresponds to a different possible output. A non-classification is when a classification system fails to select one of the known outputs and instead outputs a no-output-selected result.

[0063] Referring now to Figs. 1(a), 1(b), 1(c), and 1(d), there are shown bitmap representations of a nominal letter "O", a degraded letter "O", a nominal number "7", and a degraded letter "7", respectively. A nominal input is an ideal input with no noise associated with it. A degraded input is one in which noise has created deviations from the nominal input. Degraded inputs may result from either controlled noise or real unpredictable noise.

25

[0064] In a preferred embodiment, the training system of the present invention may train with training inputs of three different quality levels. The first level of training inputs are nominal inputs like those presented in Figs. 1(a) and 1(c). The second level of training inputs are controlled noise inputs, a type of degraded input created by applying defined noise functions or signals with different characteristics, either independently or in combination, to nominal inputs. The third level of training inputs are real noise inputs, a second type of degraded inputs which, in the case of characters, may be optically acquired images of known characters. Such degraded inputs have real unpredictable noise. Figs. 1(b) and 1(d) present representations of possible controlled noise inputs and real noise inputs. In a preferred embodiment, the nominal inputs have the highest quality, with the controlled noise inputs and real noise inputs of decreasing lesser quality. Depending upon the controlled noise functions and signals applied, a particular controlled-noise input may be of greater or lesser quality than a particular real-noise input.

35

[0065] The quality of a particular degraded input -- of either controlled-noise or real-noise variety -- may be determined by comparing the degraded input to a nominal input corresponding to the same known character. In a preferred embodiment, a quality measure may be based on the number of pixels that differ between the two inputs. In another preferred embodiment, the quality measure may be based on conventional feature measures such as Grid or Hadamard features.

40

[0066] In a preferred embodiment, training systems of the present invention train first with the nominal inputs and then later with degraded controlled-noise and real-noise inputs. In this preferred embodiment, training with inputs corresponding to Figs. 1(a) and 1(c) would precede training with those of Figs. 1(b) and 1(d). In another preferred embodiment, the training system trains with all inputs of the same known character prior to proceeding to the next known character, and the training inputs of each known character are internally organized by quality. In this preferred embodiment, training with Fig. 1(a) proceeds that with Fig. 1(b), and training with Fig. 1(c) proceeds that with Fig. 1(d). Those skilled in the art will understand that the exact overall sequence of training with all of the inputs is of lesser importance than ordering of inputs by quality for each different known character.

50

REFINEMENT OF NEURONS

[0067] After the training system of the present invention has completed training, the feature space is populated with neurons that encompass feature vectors, with one feature vector corresponding to each distinct training input. Each neuron may encompass one or more feature vectors -- the one at the center of the neuron that was used to create the neuron and possibly other feature vectors corresponding to inputs associated with the same known character.

55

[0068] Depending upon the quality ordering of the training inputs used in the sequential training, a particular neuron may encompass those feature vectors in a more or less efficient manner. For example, if the feature vector used to create a particular neuron corresponds to a highly degraded input, then that feature vector will lie at the center of that

neuron. That same neuron may also encompass other feature vectors corresponding to nominal inputs and inputs of lesser degradation. Such a neuron may not be the most efficient neuron for encompassing that set of feature vectors. A classification system using such a neuron may make more misclassifications and non-classifications than one using a more efficient neuron.

[0069] A refinement system of the present invention spatially adjusts neurons, created during training, to create more efficient neurons. This refinement system may characterize the spatial distribution of feature vectors encompassed by a particular neuron and then spatially adjust that neuron. Such spatial adjustment may involve translating the neuron from its current center point toward the mean of the spatial distribution of those feature vectors. After translating the neuron, the axis lengths may be adjusted to ensure that feature vectors of the same output character are encompassed by the neuron and to ensure that feature vectors of different output character are excluded.

[0070] In an alternative embodiment, the refinement system may spatially adjust two or more neurons of the same character to create one or more neurons that more efficiently encompass the same feature vectors, where a feature vector from one original neuron may be encompassed by a different more efficient neuron. For example, before refinement, a first neuron may encompass feature vectors F_1 , F_2 , and F_3 , and a second neuron may encompass feature vectors F_4 , F_5 , F_6 and F_7 . After refinement, feature vectors F_1 , F_2 , F_3 and F_4 may be encompassed by a third neuron, and feature vectors F_5 , F_6 and F_7 may be encompassed by a fourth neuron, where the centers and axis lengths of the third and fourth neurons are all different from those of the first and second neurons.

CLASSIFYING SYSTEMS WITH CLUSTER CLASSIFIERS

[0071] In a first preferred embodiment of the present invention, a classification system classifies inputs into one of a set of possible outputs by comparing the feature vector, for each input to be classified, with every neuron in the feature space. Such classification systems are presented in Figs. 3 and 4.

[0072] Referring now to Fig. 6, there is shown classification system 600 -- a second preferred embodiment of the present invention -- in which inputs are classified into one of a set of possible outputs using neurons and cluster classifiers. Classification system 600 includes top-level classifier 602 and two or more cluster classifiers 604, 606, ..., 608. Top-level classifier 602 classifies inputs into appropriate clusters of inputs. For example, where classification system 600 classifies characters, top-level classifier 602 may classify input bitmaps corresponding to optically acquired characters into clusters of characters.

[0073] The characters clustered together may be those represented by similar bitmaps, or, in other words, those characters associated with feature vectors close to one another in feature space. For example, a first character cluster may correspond to the characters "D", "P", "R" and "B". A second character cluster may correspond to the characters "O", "C", "D", "U", and "Q". A third cluster may correspond to only one character such as the character "Z". A particular character may be in more than one character cluster. In this example, the character "D" is in both the first and the second character clusters, because its bitmaps are similar to those of both clusters.

[0074] In a preferred embodiment, before training, characters are clustered based on a confusion matrix. The confusion matrix represents the likelihood that one character will be confused with another character for every possible pair of characters. In general, the closer the feature vectors of one character are to those of another character, the higher the likelihood that those two characters may be confused. For example, the character "D" may have a higher confusion likelihood with respect to the "O" than to the "M", if the feature vectors for "D" are closer to the feature vectors for "O" than to those for "M".

[0075] In a preferred embodiment, the clustering of characters is based upon a conventional K-Means Clustering Algorithm, in which a set of templates is specified for each character, where each template is a point in feature space. The K-Means Clustering Algorithm determines where in feature space to locate the templates for a particular character by analyzing the locations of the feature vectors for all of the training inputs corresponding to that character. Templates are preferably positioned near the arithmetic means of clusters of associated feature vectors.

[0076] In a preferred embodiment, four templates may be used for each character and the number of characters per cluster may be roughly even. For example, when classifying the 64 characters corresponding to the 26 capital and 26 lower-case letters, the 10 digits, and the symbols "&" and "#", 4x64 or 256 templates may be used to define 7 different clusters of roughly equivalent numbers of characters.

[0077] By clustering characters, top-level classifier 602 may implement a classification algorithm that quickly and accurately determines the appropriate cluster for each input. In a preferred embodiment, top-level classifier 602 implements a neuron-based classification algorithm. In another preferred embodiment, other conventional non-neural classification algorithms may be performed by top-level classifier 602. Top-level classifier 602 selects the appropriate cluster for a particular input and directs processing to continue to the appropriate cluster classifier 604, 606, ..., 608. Each cluster classifier is associated with one and only one character cluster, and vice versa.

[0078] In one preferred embodiment, each cluster classifier may implement a classification algorithm unique to that character cluster, or shared by only a subset of the total number of character clusters. Each cluster classifier may

therefore employ neurons that exist in a feature space unique to that character cluster. For example, training for the "P", "R", "B" cluster may employ a particular set of Grid features, while training for the "O", "C", "D", "U", "Q" cluster may employ a different set of Hadamard features. In that case, different training procedures are performed for each different cluster classifier, where only inputs corresponding to those characters of the associated cluster are used for each different training procedure.

[0079] In a third preferred embodiment of the present invention, a classification system according to Fig. 8 may classify inputs into one of a set of possible outputs using neurons and cluster classifiers. In this third embodiment, top-level classifier 602 identifies the template in feature space closest to the feature vector for the current input to be classified. The identified template is associated with a particular character that belongs to one or more character clusters. The top-level classifier 602 directs processing to only those cluster classifiers 604, 606, ..., 608 associated with the character clusters of the closest template. Since a particular character may be in more than one character cluster, more than one cluster classifier may be selected by top-level classifier 602 for processing.

[0080] In a fourth preferred embodiment, each cluster classifier may have a decision tree that identifies those neurons that should be processed for a given input. Prior to classifying, feature vector space for a particular cluster classifier may be divided into regions according to the distribution of feature vectors and/or neurons in feature space. Each region contains one or more neurons, each neuron may belong to more than one region, and two or more regions may overlap. Top-level classifier 602 may determine in which feature-space region (or regions) the feature vector for the current input lies and may direct the selected cluster classifiers to process only those neurons associated with the region (or those regions).

[0081] Those skilled in the art will understand that some classification systems of the present invention may use decision trees without cluster classifiers, some may use cluster classifiers without decision trees, some may use both, and others may use neither. Those skilled in the art will further understand that decision trees and cluster classifiers may increase the efficiency of classification systems of the present invention by reducing processing time.

25 PREFERRED AND ALTERNATIVE PREFERRED EMBODIMENTS

[0082] Those skilled in the art will understand that classifying systems of the present invention may be arranged in series or parallel. For example, in a preferred embodiment, a first character classifier based on Grid features may be arranged in series with a second character classifier based on Hadamard features. In such case, the first classifier classifies a particular bitmap input as one of the known characters or it fails to classify that input. If it fails to classify, then the second classifier attempts to classify that input.

[0083] In an alternative embodiment, two or more different classifiers may be arranged in parallel. In such case, a voting scheme may be employed to select the appropriate output by comparing the outputs of each different classifier.

[0084] In a preferred embodiment, classification systems and training systems of the present invention perform parallel processing, where each elliptical processing unit may run on a separate computer processor during classification, although those skilled in the art will understand that these systems may also perform serial processing. In a preferred embodiment, the classification systems and training systems may reside in a reduced instruction set computer (RISC) processor such as a SPARC 2 processor running on a SPARCstation 2 marketed by Sun Microsystems.

[0085] Those skilled in the art will understand that inputs other than character images may be classified with the classification systems of the present invention. In general, any input may be classified as being one of a set of two or more possible outputs, where a no-selection result is one of the possible outputs. For example, the classification systems of the present invention may be used to identify persons based-upon images of their faces, fingerprints, or even earlobes. Other classification systems of the present invention may be used to identify people from recordings of their voices.

[0086] It will be further understood that various changes in the details, materials, and arrangements of the parts which have been described and illustrated in order to explain the nature of this invention may be made by those skilled in the art without departing from the scope of the invention as expressed in the following claims.

50 Claims

1. A training method for creating a new neuron in a feature space having at least one existing neuron, comprising the steps of:
 - (a) generating a feature vector (504) representative of a training input wherein said training input corresponds to one of a plurality of possible outputs; and
 - (b) if no existing neuron corresponding to said training input encompasses said feature vector (512) then creating said new neuron;

characterised in that said new neuron comprises a boundary defined by two or more neuron axes of different length;
 said feature space comprises an existing feature vector;
 and step (b) further comprises the steps of:

- 5 (i) creating (514) a temporary neuron comprising a boundary defined by two or more temporary neuron axes;
 and
 (ii) if said temporary neuron encompasses said existing feature vector (508) and said existing feature vector does not correspond to said training input (506) then spatially adjusting said temporary neuron to create said new neuron, and in that step (b) (ii) comprises the steps of:

- 10 (1) selecting at least one of said temporary neuron axes;
 (2) calculating the distances along each of said selected temporary neuron axes from the center of said temporary neuron to said existing feature vector; and
 15 (3) reducing (510) said selected temporary neuron axes by amounts proportional to said distances and the lengths of said selected temporary neuron axes to create said new neuron.

2. The method of claim 1, wherein step (b)(ii)(1) comprises the step of selecting all of said temporary neuron axes

20 3. The method of claim 1, wherein in step (b)(ii):

- (1) only one of said temporary neuron axes is selected; and
 (2) a distance in accordance with said selected temporary neuron axis and said existing feature vector is calculated; and
 25 (3) said selected temporary neuron axis is reduced by said distance to create said new neuron.

4. The method of one of the claims 1 or 3, wherein said training input is representative of a character.

5. The method of one of the claims 1 or 3, wherein said information representative of said input comprises a feature vector.

30 6. The method of claim 5, wherein said feature vector comprises a feature element, wherein said feature element is a Grid feature element or a Hadamard feature element.

35 7. The method of one of the claims 1 or 3, wherein said neuron is a hyper-ellipse or a hyper-rectangle in a k-dimensional feature space, where k is greater than or equal to two.

8. A training apparatus for creating a new neuron in a feature space having at least one existing neuron, comprising:

40 generating means (504) for generating a feature vector representative of a training input, wherein said training input corresponds to one of a plurality of possible outputs; and
 creating means (514) for creating said new neuron, if no existing neuron (506) corresponding to said training input encompasses said feature vector;

45 characterised in that said new neuron comprises a boundary defined by two or more neuron axes of different length;
 said feature space comprises an existing feature vector;
 said creating means creates a temporary neuron comprising a boundary defined by two or more temporary neuron axes; and

50 if said temporary neuron encompasses said existing feature vector (508) and said existing feature vector does not correspond to said training input, then said creating means spatially adjusts said temporary neuron to create said new neuron; and in that said creating means

- 55 (1) selects at least one of said temporary neuron axes;
 (2) calculates the distances along each of said selected temporary neuron axes from the center of said temporary neuron to said existing feature vector; and
 (3) reduces said selected temporary neuron axes by amounts proportional to said distances and the lengths of said selected temporary neuron axes to create said new neuron.

9. The apparatus of claim 8, wherein said creating means selects all of said temporary neuron axes for said reduction.
10. The apparatus of claim 8, wherein said creating means selects only one of said temporary neuron axes, said creating means calculates a distance in accordance with said selected temporary neuron axis and said existing feature vector, and said creating means reduces said selected temporary neuron axis by said distance to create said new neuron.
11. The apparatus of claim 8, wherein said training input is representative of a character.
12. The apparatus of claim 8, wherein said information representative of said input comprises a feature vector.
13. The apparatus of claim 12, wherein said feature vector comprises a feature element, wherein said feature element is a Grid feature element or a Hadamard feature element.
14. The apparatus of claim 12, wherein said comparing means determines whether said neuron encompasses said feature vector.
15. The apparatus of claim 12, wherein said comparing means determines a distance measure from said feature vector to said neuron.
16. The apparatus of claim 8, wherein said neuron is a hyper-ellipse or a hyper-rectangle in k-dimensional feature space, where k is greater than or equal to two.

25 **Patentansprüche**

1. Trainingsverfahren zum Erzeugen eines neuen Neurons in einem Merkmalsraum, in dem sich zumindest schon ein Neuron befindet, mit den Schritten:
- 30 (a) Erzeugen eines Merkmalsvektors (504), der eine Trainingseingabe darstellt, wobei die Trainingseingabe einer von mehreren möglichen Ausgaben entspricht; und
(b) wenn kein existierendes Neuron entsprechend der Trainingseingabe den Merkmalsvektor (512) einschließt, Erzeugen des neuen Neurons,
- 35 dadurch gekennzeichnet, dass
das neue Neuron eine durch zwei oder mehr unterschiedlich lange Neuronachsen definierte Grenze aufweist;
der Merkmalsraum einen existierenden Merkmalsvektor aufweist; und
Schrift (b) außerdem die folgenden Schritte aufweist:
- 40 (i) Erzeugen (514) eines zeitweiligen Neurons mit einer durch zwei oder mehrere zeitweilige Neuronachsen definierten Grenze; und
(ii) wenn das zeitweilige Neuron den existierenden Merkmalsvektor (508) einschließt und der existierende Merkmalsvektor nicht der Trainingseingabe (506) entspricht, räumliches Anpassen des zeitweiligen Neurons,
um das neue Neuron zu erzeugen,
wobei der Schritt (b) (ii) folgende Schritte aufweist:
- 45 (1) Wählen zumindest einer der zeitweiligen Neuronachsen;
(2) Berechnen des Abstands längs jeder der gewählten zeitweiligen Neuronachsen vom Zentrum des zeitweiligen Neurons zum existierenden Merkmalsvektor; und
(3) Verkleinern (510) der ausgewählten zeitweiligen Neuronachsen um Beträge proportional zu den Abständen und den Längen der ausgewählten zeitweiligen Neuronachsen, um das neue Neuron zu erzeugen.
- 50 2. Verfahren nach Anspruch 1, bei dem der Schritt (b) (ii) (1) den Schritt des Auswählens aller zeitweiligen Neuronachsen aufweist.
- 55 3. Verfahren nach Anspruch 1, bei dem der Schritt (b) (ii) aufweist:

(1) lediglich eine der zeitweiligen Neuronachsen wird ausgewählt; und
 (2) ein Abstand nach Maßgabe der ausgewählten zeitweiligen Neuronachse und des existierenden Merkmalsvektors wird berechnet; und
 (3) die ausgewählte zeitweilige Neuronachse wird zur Erzeugung des neuen Neurons um den Abstand verkleinert.

- 5 4. Verfahren nach einem der Ansprüche 1 oder 3, bei dem die Trainingseingabe einen Buchstaben darstellt.
 10 5. Verfahren nach Anspruch 1 oder 3, bei dem die die Eingabe darstellende Information einen Merkmalsvektor aufweist.
 15 6. Verfahren nach Anspruch 5, bei dem der Merkmalsvektor ein Merkmalselement aufweist, wobei das Merkmalselement ein Gittermerkmalselement oder ein Hadamard-Merkmalselement ist.
 20 7. Verfahren nach einem der Ansprüche 1 oder 3, bei dem das Neuron eine Hyperellipse oder ein Hyperrechteck im k-dimensionalen Merkmalsraum ist, wobei $k \geq 2$ ist.
 25 8. Trainingsvorrichtung zur Erzeugung eines neuen Neurons im Merkmalsraum, der zumindest schon ein Neuron aufweist, mit:

einer Erzeugungseinrichtung (504) zum Erzeugen eines eine Trainingseingabe darstellenden Merkmalsvektor, wobei die Trainingseingabe einer von mehreren möglichen Ausgaben entspricht; und
 einer Erzeugungseinrichtung (514) zum Erzeugen des neuen Neurons, wenn kein existierendes Neuron entsprechend der Trainingseingabe den Merkmalsvektor einschließt;

30 dadurch gekennzeichnet, dass

das neue Neuron eine durch zwei oder mehrere unterschiedlich lange Neuronachsen definierte Grenze aufweist; der Merkmalsraum einen existierenden Merkmalsvektor aufweist; die Erzeugungseinrichtung ein zeitweiliges Neuron mit einer durch zwei oder mehrere zeitweilige Neuronachsen definierten Grenze erzeugt; und wenn das zeitweilige Neuron den existierenden Merkmalsvektor (508) einschließt und der existierende Merkmalsvektor nicht der Trainingseingabe entspricht, die Erzeugungseinrichtung zur Erzeugung des neuen Neurons das zeitweilige Neuron räumlich einstellt, wobei die Erzeugungseinrichtung

- 35 (1) zumindest eine der zeitweiligen Neuronachsen auswählt;
 (2) den Abstand längs jeder der ausgewählten zeitweiligen Neuronachsen von der Mitte des zeitweiligen Neurons zum existierenden Merkmalsvektor berechnet; und
 (3) die ausgewählten zeitweiligen Neuronachsen um Beträge proportional zu den Abständen und den Längen der ausgewählten zeitweiligen Neuronachsen verkleinert, um das neue Neuron zu erzeugen.
- 40 9. Vorrichtung nach Anspruch 8, bei der die Erzeugungseinrichtung alle zeitweiligen Neuronachsen für die Verkleinerung auswählt.
- 45 10. Vorrichtung nach Anspruch 8, bei der die Erzeugungseinrichtung lediglich eine der zeitweiligen Neuronachsen auswählt; einen Abstand nach Maßgabe der ausgewählten zeitweiligen Neuronachse und des existierenden Merkmalsvektors berechnet und die ausgewählte zeitweilige Neuronachse um den Abstand verkleinert, um das neue Neuron zu erzeugen.
- 50 11. Vorrichtung nach Anspruch 8, bei der die Trainingseingabe einen Buchstaben darstellt.
12. Vorrichtung nach Anspruch 8, bei der die die Eingabe darstellende Information einen Merkmalsvektor aufweist.
- 55 13. Vorrichtung nach Anspruch 12, bei der der Merkmalsvektor ein Merkmalselement aufweist, wobei das Merkmalselement ein Gittermerkmalselement oder ein Hadamard-Merkmalselement ist.
14. Vorrichtung nach Anspruch 12, bei der die Vergleichseinrichtung bestimmt, ob das Neuron den Merkmalsvektor einschließt.

15. Vorrichtung nach Anspruch 12, bei der die Vergleichseinrichtung ein Abstandsmaß vom Merkmalsvektor zum Neuron bestimmt.

16. Vorrichtung nach Anspruch 8, bei der das Neuron eine Hyperellipse oder ein Hyperrechteck im k-dimensionalen 5 Merkmalsraum ist, wobei $k \geq 2$ ist.

Revendications

10. Méthode d'apprentissage pour créer un nouveau neurone dans un espace des fonctions comprenant au moins un neurone existant, comportant les étapes suivantes:

(a) génération d'un vecteur à fonctions (504) représentatif d'une entrée d'apprentissage, ladite entrée d'apprentissage correspondant à l'une d'une pluralité de sorties possibles; et

15. (b) si aucun neurone existant correspondant à ladite entrée d'apprentissage inclut ledit vecteur à fonctions (512), alors création dudit nouveau neurone;

caractérisée en ce que

20. ledit nouveau neurone comporte une limite définie par deux axes de neurone ou plus de différentes longueurs; ledit espace des fonctions comprend un vecteur à fonctions existant; et l'étape (b) comprend en outre les étapes suivantes:

(i) création (514) d'un neurone temporaire comportant une limite définie par deux axes de neurone temporaire ou plus; et

25. (ii) si ledit neurone temporaire inclut ledit vecteur à fonctions (508) existant, et si ledit vecteur à fonctions existant ne correspond pas à ladite entrée d'apprentissage (506), alors ajustement spatial dudit neurone temporaire pour créer ledit nouveau neurone,

et en ce que l'étape (b) (ii) comporte les étapes suivantes:

30. (1) sélection d'au moins l'un desdits axes de neurone temporaire;
 (2) calcul des distances le long de chacun desdits axes de neurone temporaire sélectionnés entre le centre dudit neurone temporaire et le vecteur à fonctions existant; et
 (3) réduction (510) desdits axes de neurone temporaire par des quantités proportionnelles aux dites distances et aux longueurs desdits axes de neurone temporaire sélectionnés pour créer ledit nouveau neurone.

35. 2. Méthode selon la revendication 1, dans laquelle l'étape (b) (ii) (1) comporte l'étape de sélectionner tous lesdits axes de neurone temporaire.

40. 3. Méthode selon la revendication 1, dans laquelle à l'étape (b) (ii):

(1) seulement l'un desdits axes de neurone temporaire est sélectionné; et
 (2) une distance en accord avec ledit axe de neurone temporaire sélectionné et ledit vecteur à fonctions existant est calculée; et

45. (3) ledit axe de neurone temporaire sélectionné est réduit par ladite distance pour créer ledit nouveau neurone.

4. Méthode selon l'une des revendications 1 ou 3, dans laquelle ladite entrée d'apprentissage est représentative d'un caractère.

50. 5. Méthode selon l'une des revendications 1 ou 3, dans laquelle ladite information représentative de ladite entrée comporte un vecteur à fonctions.

6. Méthode selon la revendication 5, dans laquelle ledit vecteur à fonctions comporte un élément à fonctions, ledit élément à fonctions étant un élément à fonctions de Grid ou un élément à fonctions de Hadamard.

55. 7. Méthode selon l'une des revendications 1 ou 3, dans laquelle ledit neurone est une hyper-ellipse ou un hyper-rectangle dans un espace des fonctions k-dimensionnel, où k est supérieur ou égal à deux.

8. Appareil d'apprentissage pour créer un nouveau neurone dans un espace des fonctions comportant au moins un neurone existant, comprenant:

des moyens de génération (504) pour générer un vecteur à fonctions représentatif d'une entrée d'apprentissage, ladite entrée d'apprentissage correspondant à l'une d'une pluralité de sorties possibles; et
 5 des moyens de création (514) pour créer ledit nouveau neurone, si aucun neurone existant (506) correspondant à ladite entrée d'apprentissage inclut ledit vecteur à fonctions;

caractérisé en ce que

10 ledit nouveau neurone comporte une limite définie par deux axes de neurone ou plus de longueurs différentes; ledit espace des fonctions comporte un vecteur à fonctions existant; ledits moyens de création créent un neurone temporaire comportant une limite définie par deux axes de neurone temporaire ou plus; et si ledit neurone temporaire inclut ledit vecteur à fonctions existant (508) et si ledit vecteur à fonctions existant ne correspond pas à ladite entrée d'apprentissage, alors ledits moyens de création ajustent spatiallement ledit neurone temporaire pour créer ledit nouveau neurone;
 15 et en ce que ledits moyens de création

(1) sélectionnent au moins l'un desdits axes de neurone temporaire;
 20 (2) calculent les distances le long de chacun desdits axes de neurone temporaire sélectionnés entre le centre dudit neurone temporaire et ledit vecteur à fonctions existant; et
 (3) réduisent ledits axes de neurone temporaire sélectionnés par des quantités proportionnelles aux dites distances et aux longueurs desdits axes de neurone temporaire sélectionnés pour créer ledit nouveau neurone.

- 25 9. Appareil selon la revendication 8, dans lequel ledits moyens de création sélectionnent tous les axes de neurone temporaire pour ladite réduction.

10. Appareil selon la revendication 8, dans lequel ledits moyens de création sélectionnent seulement l'un des axes de neurone temporaire, ledits moyens de création calculent une distance en accord avec ledit axe de neurone temporaire sélectionné et ledit vecteur à fonctions existant, et ledits moyens de création réduisent ledit axe de neurone temporaire sélectionné par ladite distance pour créer ledit nouveau neurone.
 30

11. Appareil selon la revendication 8, dans lequel ladite entrée d'apprentissage est représentative d'un caractère.

- 35 12. Appareil selon la revendication 8, dans lequel ladite information représentative de ladite entrée comporte un vecteur à fonctions.

13. Appareil selon la revendication 12, dans lequel ledit vecteur à fonctions comporte un élément à fonctions, ledit élément à fonctions étant un élément à fonctions de Grid ou un élément à fonctions de Hadamard.

- 40 14. Appareil selon la revendication 12, dans lequel ledites moyens de comparaison déterminent si ledit neurone inclut ledit vecteur à fonctions.

15. Appareil selon la revendication 12, dans lequel ledits moyens de comparaison déterminent une mesure de distance entre ledit vecteur à fonctions et ledit neurone.
 45

16. Appareil selon la revendication 8, dans lequel ledit neurone est une hyper-ellipse ou un hyper-rectangle dans un espace des fonctions k-dimensionnel, où k est supérieur ou égal à deux.

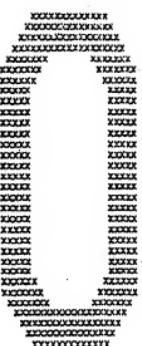
50

55

1(a)

Fig.

Nominal "0"



1(b)

Fig.

Degraded "0"



1(c)

Fig.

Nominal "7"

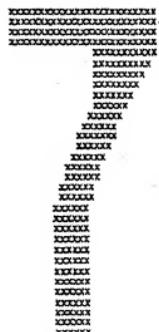


Fig.

Degraded "7"



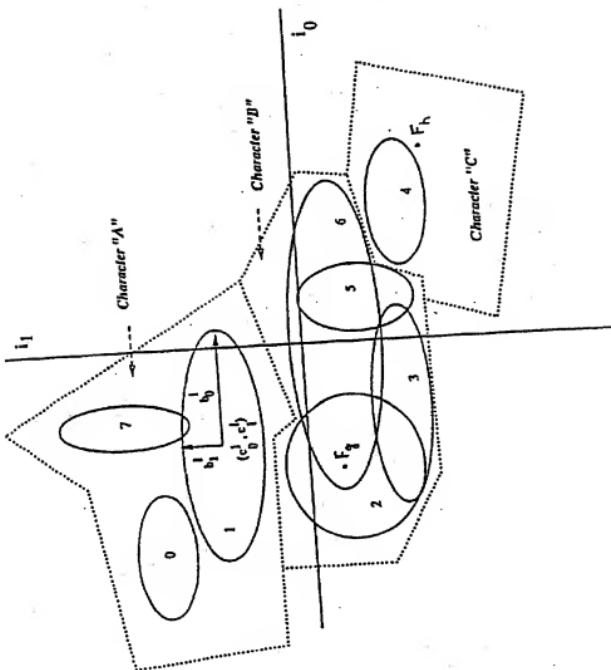
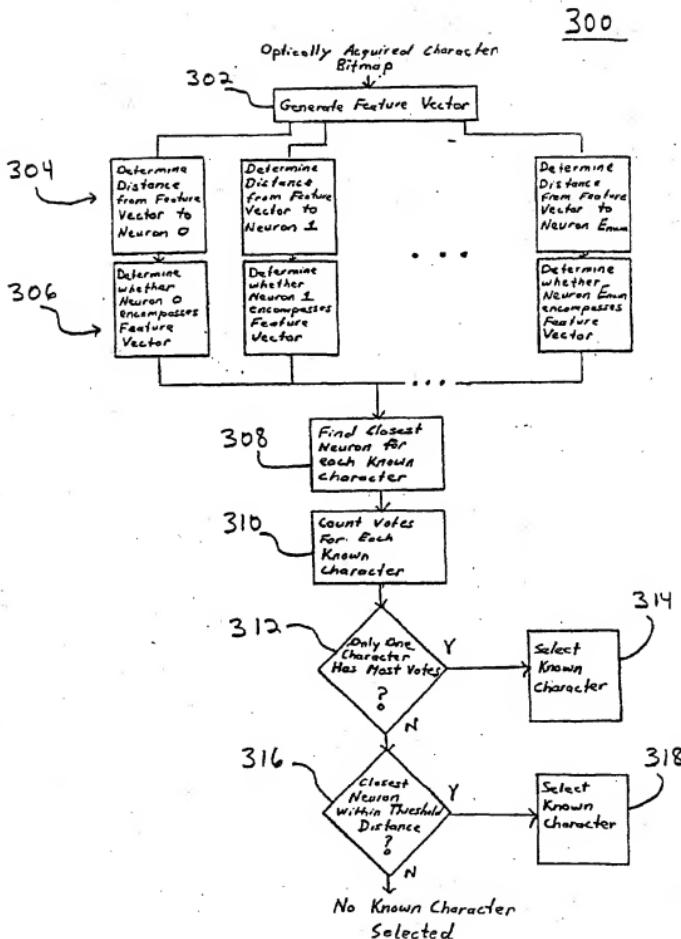


Fig. 2

Figure 3.



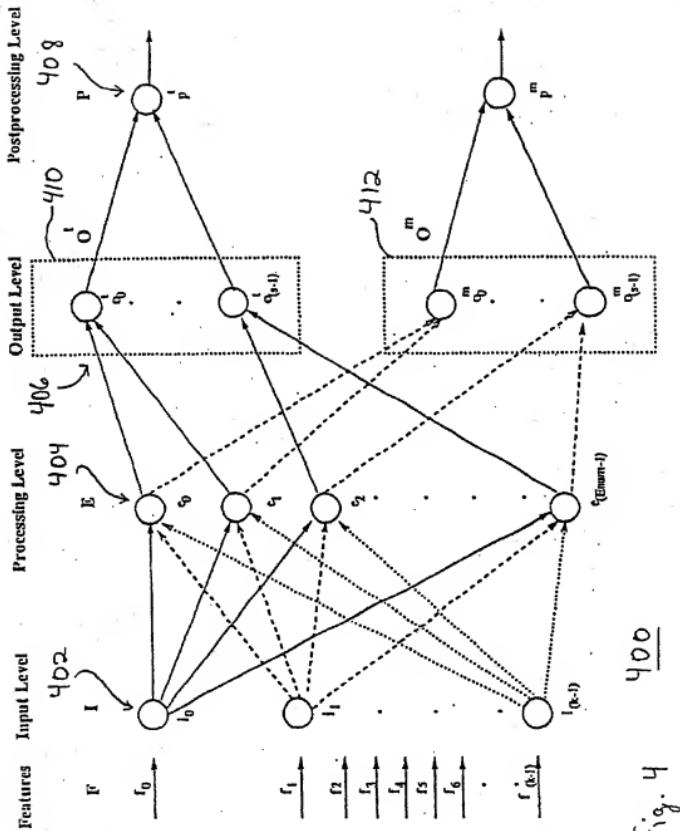


Fig. 4

Figure 5

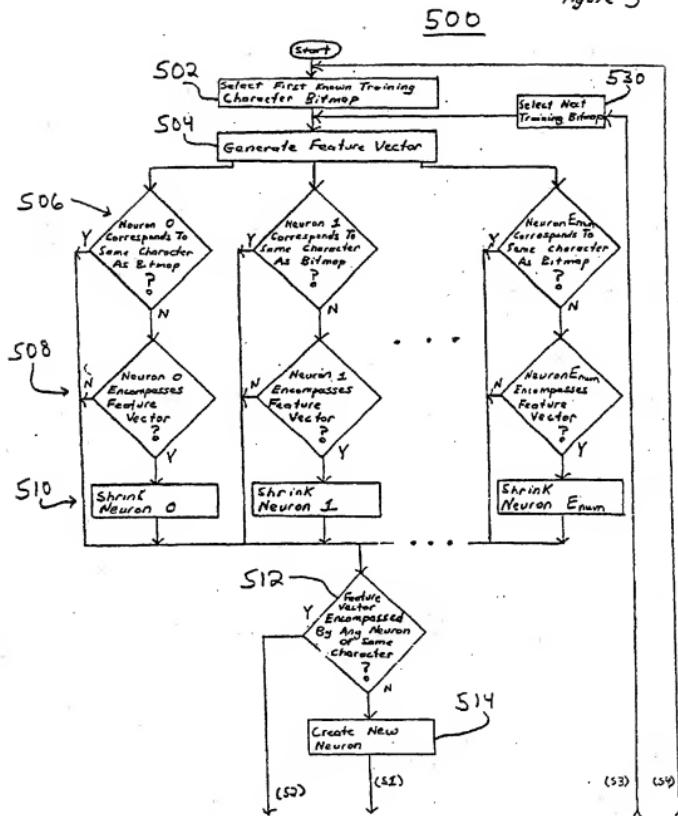


Figure 5 (continued)

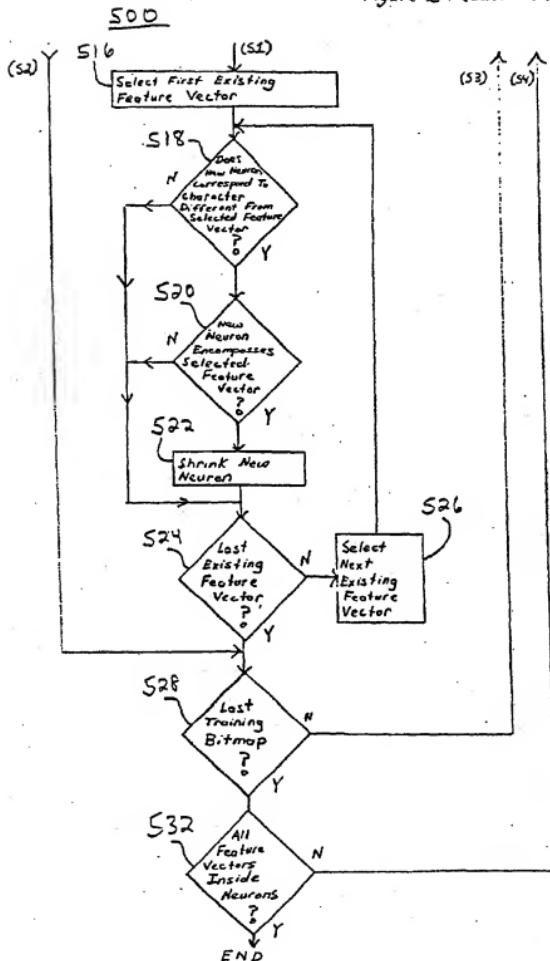


Fig. 6

